Amendments to the Claims:

1. (currently amended) A method for generating test scripts comprising:

~~Storing test cases in abstract representations to generate test cases in any target environment script format to provide interoperability between automation tools and cross environment portability of test cases;~~

providing at least one particular test case in a data store;

using semantic analysis to decompose the at least one particular test cases into ~~application states~~ a platform-independent abstract representation which comprises at least three components, and wherein the at least three components include at least one application state, ~~applications~~, at least one external interaction sequences~~,~~ and input data ~~without changing or deleting an original test case~~ ;

generating ~~rule-based~~ at least one rule-based test cases ~~using rule-based generation of test cases from an~~ based on the abstract representation ~~that includes application states, external interaction sequences and input data of test cases from data stores, the test cases being recombined and modified using external rules to combine and modify components of the abstract representation of test cases into new scripts~~, wherein ~~each~~ the at least one application state ~~is~~ comprises at least one of: (a) a set of application objects associated with a set of attributes and their values, or (b) ~~represents~~ a runtime snapshot of an application under test which defines a context of external interaction;

validating the at least one rule-based test cases; and

~~converting~~ generating at least one test script based on the at least one rule-based test cases~~,~~ ~~to test scripts that are platform independent and the rule based test cases can be used to generate scripts in different~~ wherein the at least one test script can be run in a particular one of a plurality of target environments.

2. (currently amended) The method of claim 1, wherein a the data store is a relational database management system.

3. (currently amended) The method of claim 1, wherein a the data store is an XML database management system.

4. (currently amended) The method of claim 1, wherein a the data store is a file system.

5. (currently amended) The method of claim 1, wherein an the at least one application state ~~represents~~ comprises a representation of a runtime snapshot of application under test which defines the context of external interaction.

6. (currently amended) The method of claim 5, wherein the at least one application state ~~includes~~ comprises a set of application objects, ~~its~~ attributes of the application objects, and values of the attributes ~~values~~.

7. (currently amended) The method of claim 5, wherein the at least one application state is a plurality of application states, and the plurality of application states ~~corresponding to a test case~~ are arranged in a hierarchical manner.

8. (currently amended) The method of claim 1, wherein the at least one external interaction sequences ~~represent~~ comprises a representation of events invoked by at least one external agents on the set of application objects.

9. (currently amended) The method of claim 8, wherein the at least one external agents ~~are~~ is at least one of a human agents or ~~other~~ a software agents.

10. (currently amended) The method of claim 8, wherein the at least one interaction ~~sequencing includes~~ sequence comprises at least one flow control structures for capturing at least one of a sequential interaction, a concurrent interaction, a looping interaction, or a ~~and~~ conditional interactions.

11. (currently amended) The method of claim 1, wherein the ~~validation of generated test cases includes~~ validating step comprises both internal validation and external validation.

12. (currently amended) The method of claim 11, wherein the internal validation ensures that ~~the~~ components of the at least one particular test case ~~definition~~, the at least one external interaction sequences, and the input data are consistent with each other and with an application object model.

13. (currently amended) The method of claim 12, wherein ~~an~~ the application object model ~~is~~ comprises a metadata representation for modeling an application under test.

14. (currently amended) The method of claim 13, wherein the metadata representation ~~includes~~ comprises object type definitions for application objects.

15. (currently amended) The method of claim 13, wherein the metadata representation ~~includes~~ comprises attribute definitions for each type of application object ~~type~~.

16. (currently amended) The method of claim 13, wherein the metadata representation ~~includes~~ comprises a definition of methods and events that are supported by each type of application object ~~type~~.

17. (currently amended) The method of claim 13, wherein the metadata representation ~~includes~~ comprises a definition of effects of events on ~~an~~ the at least one application state.

18. (currently amended) The method of claim 14, wherein ~~application~~ the object type definitions ~~include~~ further comprise ~~additional~~ categorization of each type of application object ~~types into~~ as a hierarchical type, a container types, ~~and~~ or a simple type~~s~~.

19. (currently amended) The method of claim 18, wherein ~~the~~ a particular application state is associated with each hierarchical object type~~s are associated with an application state of its own~~, and wherein a container type is an application object type~~s~~ that can contain ~~instances of~~ other application objects ~~are termed container types~~.

20. (currently amended) The method of claim 19, wherein the particular application state associated with a hierarchical ~~application~~ object type is at least one of a modal application state or a nonmodal application state.

21. (currently amended) The method of claim 20, wherein a the modal application state restricts possible interactions to application object instances available within the ~~current~~ at least one application state.

22. (currently amended) The method of claim 17, wherein the effects of events on an application state capture one or more consequences of the event to the application state.

23. (currently amended) The method of claim 22, wherein a consequence of an event is selected from~~,~~ the set consisting of: creation of a new object instance of a given type, deletion of an object instance of a given type, modification of attributes of an existing object instance, and selection of an instance of an object type.

24. (currently amended) The method of claim 23, wherein creation of a new object instance of ~~an object of type that is~~ a hierarchical object type results in creation of a new application state.

25. (currently amended) The method of claim 23, wherein the selection of an object instance of ~~type that is~~ a hierarchical object type results in selection of ~~the~~ a certain application state associated with that object instance.

26. (currently amended) The method of claim 11, wherein the external validation validates the at least one rule-based generated test case against ~~the~~ an application metadata repository.

27. (currently amended) The method of claim 26, wherein the application metadata repository ~~contains~~ comprises definitions of application objects and ~~nature of their~~ specification of interactions of the application objects within the application under test.

28. (currently amended) The method of claim 26, wherein the external validation ~~serves as~~ provides a static verification test for the at least one rule-based generated test cases.

29. (currently amended) The method of claim 26, wherein the external validation increases productivity by ~~pointing out~~ identifying invalid test cases.

30. (currently amended) The method of claim 26, wherein the external validation increases productivity by ~~pointing out~~ identifying inconsistencies in statically verifiable application behaviors.

31. (currently amended) The method of claim 1, wherein the at least one test scripts ~~are~~ comprises a rule-based test cases represented in a scripting language.

32. (currently amended) The method of claim 31, wherein the scripting languages ~~can be~~ is at least one of a typed programming language or an untyped programming languages used for at least one of recording or authoring test cases.

33. (currently amended) The method of claim 1, further comprising:

providing rules for selection of components of test case definitions, external interaction sequences and input data; and further providing rules for data driven test case generation.

34. (currently amended) The method of claim 33, wherein the selection rules are specified using at least one query language~~s~~.

35. (currently amended) The method of claim 34, wherein the at least one query language is Structured Query Language (SQL).

36. (currently amended) The method of claim 34, wherein the at least one query language is Extensible Markup Language ~~(XML)~~ Query (XQuery) language.

37. (currently amended) The method of claim 34, wherein the at least one query language is Application Programming Interface (API) called from code written in a programming language.

38. (currently amended) The method of claim 34, wherein the use of the at least one query languages allows test cases to be generated from live customer data.

39. (currently amended) The method of claim 33, wherein the data driven test case generation ~~involves~~ comprises composing the test case ~~as dictated by~~ based on the input data.

40. (currently amended) The method of claim 39, wherein the ~~availability of multiple datasets for the~~ input data comprises a plurality of datasets, and ~~will result in generation of multiple~~ wherein at least one of a plurality of test cases or a plurality of external interaction sequences repeated within a loop control structure is generated for each dataset.

41. (currently amended) The method of claim 39, wherein ~~the availability of multiple datasets for a~~ portion of the input data comprises a plurality of datasets, and ~~will result in~~ the interaction sequences corresponding to ~~this~~ the portion of input data are repeated within a loop control structure.

42. (currently amended) The method of claim 39, wherein each element of input data ~~can be~~ is flagged as either valid or invalid.

43. (currently amended) The method of claim 42, wherein the generation step further comprises providing an appropriate interaction sequence for exception handling when the presence of a first validity flag in an element of the input data that is different from ~~the one~~ a second validity flag corresponding to the input data when the at least one particular test cases was at least one of recorded or authored~~, results in the generator including appropriate interaction sequences for exception handling~~.

44. (currently amended) The method of claim 1, further comprising:

using a language mapping to generate the at least one test script in at least one scripting language ~~converting test case from internal representation to a scripting language through language mapping~~.

45. (currently amended) The method of claim 44, wherein the language mapping ~~is used to~~ maps external interactions captured as events on an application object to ~~appropriate~~ particular statements in the scripting language.

46. (currently amended) The method of claim 44, wherein ~~more than one~~ a plurality of language mappings are provided at the same time.

47. (currently amended) The method of claim 44, wherein ~~the generated test case are~~ a plurality of test scripts is converted ~~to more than one~~ using a plurality of scripting languages at the same time.

48. (currently amended) The method of claim 47, wherein ~~generating test cases in multiple scripting language allows generation of test scripts for multiple~~ the plurality of test scripts can be used in a plurality of test execution environments.

49. (currently amended) A computer system, comprising:

a processor[[:]] ;

a memory arrangement coupled to the processor, the memory ~~storing~~ arrangement configured to store at least one particular test case; ~~rule-based test cases from an abstract representation that includes application states, external interaction sequences, and input data of test cases from data stores to produce test cases, the memory storing test cases in abstract representations to generate test cases in any target environment script format to provide interoperability between automation tools and cross-environment portability of test cases~~;

~~logic that validates the rule based test cases; and~~

~~logic~~ a first set of instructions which, when executed by the processor, configures the processor to ~~that~~ uses semantic analysis to decompose the at least one particular test cases into ~~applications external interaction sequences and input data~~ a platform-independent abstract representation which comprises at least three components, and wherein the at least three components include at least one application state, at least one external interaction sequence, and input data;

a second set of instructions which, when executed by the processor, configures the processor to ~~and~~ generates ~~the~~ at least one rule-based test cases ~~using rule based generation of test cases from an~~ based on the abstract representation ~~that includes application states, external interaction sequences and input data of test cases from data stores without changing or deleting an original test case~~, wherein each application state

~~is~~ comprises at least one of: (a) a set of application objects associated with a set of attributes and their values, or (b) ~~represents~~ a runtime snapshot of an application under test which defines a context of external interaction;

a third set of instructions which, when executed by the processor, configures the processor to validate the at least one rule-based test case; and

a fourth set of instructions which, when executed by the processor, configures the processor to generate at least one test script based on the at least one wherein rule-based test ~~cases, are converted to test scripts that are platform independent and the rule based test cases can be used to generate~~ scripts in different wherein the at least one test script can be run in a particular one of a plurality of target environments~~, wherein the test cases are recombined and modified using external rules to combine and modify components of the abstract representation of test cases into new scripts.~~

50. (currently amended) The system of claim 49, wherein the ~~logic that validates the test cases~~ third set of instructions, when executed by the processor, further configures the processor to verify ~~provides~~ that components of ~~a test case definition~~ the at least one test script, the at least one external interaction sequences~~,~~ and the input data are each consistent with each other and with an application object model.

51. (currently amended) The system of claim 49, wherein the ~~logic that validates the test cases~~ third set of instructions uses ~~is~~ external validation logic.

52. (currently amended) The system of claim 51, wherein the ~~external validation logic includes~~ third set of instructions, when executed by the processor, further configures the processor to ~~validating~~ validate ~~a generated test case~~ the at least one test script against an application metadata repository.

53. (currently amended) The system of claim 49, further comprising:

~~logic for providing~~ a fifth set of instructions which, when executed by the processor, configures the processor to provide rules for selection of components of test case definition, external interaction sequences and input data; wherein and the rules are data driven test case generation.

54. (currently amended) The system of claim 49, further comprising:

~~logic for providing~~ a sixth set of instructions which, when executed by the processor, configures the processor to provide data driven test case generation.

55. (currently amended) The system of claim 54, wherein the ~~logic for providing data driven test case generation includes composing~~ sixth set of instructions, when executed by the processor, further configures the processor to compose the at least one rule-based test case as dictated by the input data.